An Intrusion Detection module using Neural Networks

For use in the Anomalous system

Dylan Brown
Supervised by:
Andrew Hutchison
Michelle Kuttel
10/22/2012

<table>
<thead>
<tr>
<th>ITEM</th>
<th>Given</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements Analysis and Design</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>Theoretical Analysis</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Experiment Design and Execution</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>System Development and Implementation</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Results, Findings and Conclusion</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Aim Formulation and Background Work</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>Quality of Report Writing and Presentation</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Adherence to Project Proposal and Quality of Deliverables</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Overall General Project Evaluation</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Total Marks</td>
<td>80</td>
<td></td>
</tr>
</tbody>
</table>
Abstract

With an increasing reliance on computer networks for daily operations of many businesses, it is important for business operators to protect themselves from network based intrusion. We propose an Intrusion Detection module for a full package system (Anomalous) which can provide classification of network based intrusions. The design of the ID module seeks to satisfy various criteria required by the Anomalous system and the context in which it is used. The design we propose is a Neural Network based solution, using backpropagation and NSL-KDD data set.
Table of Contents

1 Introduction .................................................................................................................. 4
2 Anomalous system ......................................................................................................... 4
3 Background .................................................................................................................... 5
  3.1 Misuse detection ....................................................................................................... 6
  3.2 Specification detection ............................................................................................. 6
  3.3 Anomaly detection ................................................................................................... 6
4 ID Module ....................................................................................................................... 9
  4.1 Requirements ........................................................................................................... 9
5 ID Module Design ......................................................................................................... 10
  5.1 Context .................................................................................................................... 10
  5.2 Reasons for choosing Neural Network based approach ......................................... 11
  5.3 Neural Network Design ........................................................................................ 12
    5.3.1 Inputs ............................................................................................................... 12
    5.3.2 Hidden Layer .................................................................................................. 14
    5.3.3 Output ............................................................................................................ 14
  5.4 Learning ................................................................................................................... 15
  5.5 Detection ................................................................................................................. 16
  5.6 Code Structure ...................................................................................................... 16
  5.7 Evaluation .............................................................................................................. 16
  5.8 Implementation .................................................................................................... 17
  5.9 Integration ............................................................................................................. 18
6 Discussion ..................................................................................................................... 18
7 Conclusions .................................................................................................................. 19
8 Future work .................................................................................................................... 20
9 Acknowledgements ..................................................................................................... 21

Works Cited ..................................................................................................................... 22
Appendix A ...................................................................................................................... 24
1 Introduction

Data integrity and security are increasing concerns in this data centric age we find ourselves in (Garcia-Teodoro, et al., 2009). As a result of the increased occurrences of unauthorised access to sensitive data (Kumar, et al., 1994) Intrusion Detection Systems (IDS) are becoming increasingly important for both prevention and analysis of network intrusions. Various intrusion detection approaches exist, ranging from statistically based approaches, through knowledge based approaches (whereby detection is based off a set of predefined rules), to machine learning based approaches (Debar, et al., 1999).

Intrusion detection can be used in varying circumstances, ranging from ad-hoc networks (Tseng, et al., 2006) to analysis of security events (such as login events). In each circumstance the underlying methods used are similar in nature but with varying additions and augmentations to allow for optimisation for the specific problem. These can vary from different sets of rules in knowledge based systems to different choices in machine learning based techniques such that the technique fits the requirements and constraints of the system to which it will be applied (Garcia-Teodoro, et al., 2009).

This project seeks to design an Intrusion Detection (ID) module for use in a full package (Collection, Detection and Analysis) system called Anomalous. The scope is thus limited to design of the system and not inclusive of the actual development.

In section 2 we will look at the Anomalous system and the role the each module plays within the system. In section 3 we take a detailed look at a variety of ID solutions and analyse their strengths and weakness. We must then, in section 4, look at the requirement of the ID module. With requirements in mind we can look to section 5 for the proposed design. Section 6 then follows with a discussion of the design. In section 7 we draw some conclusions on the design before looking at possible future improvements in section 8.

2 Anomalous system

The Anomalous system provides collection of data, ID based on collected data, and facilitates analysis of the target network through an interactive visualisation. This allows for a full network security solution which can be deployed in any environment to provide security for any target network. Clients of the system could range from banks and governments, to retail stores and schools.

![Figure 1 The structure of the Anomalous system](image-url)
The tasks of the *Anomalous* system are achieved by three separate modules which when integrated form the complete system.

The Collection module collects and aggregates the event data from the target network and stores the data in a database for access by the other module.

The ID module uses the collected data to detect attack behaviour and stores a list of suspicious events.

The list of suspicious events as well as the raw data is used by the Visualisation module to provide system administrators with a visual means of interpreting the results of the ID and state of the network. This provides a more efficient means of interpreting the data than current log based displays which require the system administrators to browse through textual logs.

The *Anomalous* system is structured such as to facilitate smooth communication between the modules and provide a system with efficient performance. The modules only communicate with each other when it is required by the system administrator.

The system administrator would request the Visualisation module to display results from a certain time frame. The data for this time frame would also be collected from the Collection module database through the use of an API provided by the collection module. A request would also be sent to the ID module for anomalies within the given time frame.

Using the Collection module’s data the ID module would determine the anomalies and store this data in a database for access by the Visualisation module. Once the ID module is finished the Visualisation module would be notified.

The Visualisation module would then display the data from the Collection and ID modules for analysis by the system administrator.

An illustration of the *Anomalous* system’s structure and communication can be seen in the following figures.

![Figure 2 Communication between the various modules, originating from a request through the visualisation module](image)

![Figure 3 Flow of data within the *Anomalous* system](image)

### 3 Background

The ID module is responsible for the detection of attacks within the target network. This is a common problem which has seen many different solutions and various approaches across the years.

There are three major types ID approaches, misuse detection, specification detection and anomaly detection (Debar, et al., 1999).
3.1 Misuse detection
Misuse detection is the method of detecting anomalies based on predefined signatures. Misuse detection attempts to encode knowledge about attacks as well as defined patterns and monitors for the occurrence of these patterns. This technique specifically represents knowledge about unacceptable behaviour and attempts to detect its occurrence. According to reports from response teams such as CERT the majority of intrusions come from a small number of known attacks, as such misuse detection works well in practice due to the high frequency of these known attacks (Kumar, et al., 1994).

Techniques such as pattern matching (Kumar, et al., 1994) are often used in misuse detection. Structures such as Coloured Petri Nets can be used for state transition analysis in misuse detection (Jensen, 1992).

Misuse detection relies upon attack profiles which are manually crafted by system experts. This process can be time consuming and requires expert knowledge of the system and the types of attacks it is likely to face. Using known attack profiles also prevents misuse detection from being able to detect novel attacks.

3.2 Specification detection
Unlike misuse detection, specification detection and anomaly detection are used to detect deviations from normal behaviour.

Specification detection relies on given specifications that capture legitimate system behaviour. Due to this, specification detection is highly applicable in the case of network based intrusion detection due to the well defined network specifications (Tseng, et al., 2006). For systems without well defined specifications manual intervention is required by a system expert who is specialised in understanding the behaviour of the system under non-attack conditions.

Specification detection has been applied to target networks with well defined protocols, such as mobile ad-hoc networks (MANET).

Tseng et al (Tseng, et al., 2006) propose a specification based intrusion detection model for detecting attacks on routing protocols in MANETs. The proposed specification based approach analyses the protocol specification (e.g., RFC) of an ad hoc routing protocol to establish a finite-state-automata (FSA) model that captures the correct behaviour of nodes supporting the protocol. The intrusion detection problem is therefore reduced to monitoring of the individual nodes in the MANET for violation for the constraints. Cooperative distributed detectors can then be used for the intrusion detection monitoring.

The major drawback of this approach is that the specifications needed are often system dependent and without the existence of well established specifications, constructing the normal behaviour specifications can be time consuming (Sekar, et al., 2002) (Tseng, et al., 2006).

3.3 Anomaly detection
Garcia-Teodoro et al (Garcia-Teodoro, et al., 2009) provide a summary of the various anomaly based intrusion detection techniques. All of the anomaly based systems follow a certain functional architecture (as seen below). In the parameterisation stage the observed behaviour of the monitored system is represented in a pre-established form. The normal behaviour of the system is then characterised in the training stage in order to build a model of the normal system behaviour. In the detection stage the model of the system is compared with the observed behaviour and if deviations within a given
threshold are detected then a flag will be raised.

Various approaches to this statistical based technique exist. The model used could be of the univariate, multivariate or time series types. Each model holds its own pros and cons. All the models share some features inherent in all statistical approaches.

Statistical approaches do not require prior knowledge of the systems normal activity. These statistical approaches can provide accurate notification of malicious activities occurring over long periods of time. Drawbacks to this statistical based anomaly detection include the fact that this kind of IDS can be trained by attackers in such a way that the behaviour generated during an attack is considered normal. Setting the values of the different parameters/metrics can be a difficult task, especially when trying to achieve an accurate IDS. Assumptions are also made as to the applicability of the various statistical modelling techniques.

Machine learning based techniques are a common and useful approach to building IDSs. There exists a large number of different machine learning based techniques. In general these techniques seek to use labelled data to train the behavioural model, which is then used to categorize the patterns analysed. There are many different machine learning techniques used for anomaly detection (Garcia-Teodoro, et al., 2009).

Bayesian networks can be used as one approach for machine learning based IDS. A Bayesian network is a model that encodes probabilistic relationships among variables of interest. This technique is most commonly used in conjunction with statistical schemes. This combination yields several advantages (Heckerman, 1995), such as the ability to incorporate prior knowledge and data. The results (Kreugal, et al., 2003) are often similar to those derived from threshold based systems, whilst still requiring the high
computational effort of machine learning systems. Bayesian networks have proved to be successful in certain situations; however the results obtained are highly dependent on the assumptions about the behaviour of the target system. A deviation in these hypotheses therefore leads to an increased amount of detection errors.

Techniques based on real life observations of biological systems can also be used in anomaly detection. Genetic algorithms are based on observed behaviour in evolutionary biology such as mutation, selection, inheritance and recombination. This technique can therefore derive classification rules, select appropriate features or optimal parameters for the detection process. The main advantage with this form of machine learning is that it is highly flexible and robust and it converges to an optimal solution from multiple directions whilst no prior knowledge of the system behaviour is required. The major drawback with this approach is the extra computational power required by the technique and therefore the increased time required for learning (Garcia-Teodoro, et al., 2009). Another drawback is that the system’s performance may not be optimal if the learning process does not find the correct solution (Bridges, et al., 2000).

Another commonly used machine learning technique which is applicable for use in IDS is neural networks. Neural networks (NN) attempt to emulate the operation of the human brain. These techniques have been adopted in anomaly based IDSs because of the flexibility and adaptability to environmental changes. This detection approach can be used to predict the next command from a sequence of previous ones, identify intrusive behaviour patterns and create user profiles. The major downfall is that they do not provide a clear description of why the detection decision has been made (Garcia-Teodoro, et al., 2009).

NN consist of a number of nodes (or neurons) which are divided into various layers (Jain, et al., 1996). NN used in ID usually consist of three layers: the input layer, the hidden layer and the output layer. The nodes in each layer are connected by weights. The number of nodes per layer and the connections between them are design choices, which vary across different implementations (Mukkamala, et al., 2002). A common type of NN is a feed forward NN which is illustrated by figure 5.

In a feed forward NN the connections between the nodes do not form a directed cycle. The information moves in only one direction and there are no loops in the network (Jain, et al., 1996).

Figure 5 Example of a feed forward NN

The learning process for NN determines the values of the weights. These values are then used combined with the neuron values and used to determine the output of the network during detection. The ability of a NN to learn from examples makes it an attractive solution (Jain, et al., 1996).

NN can be applied to ID through many different techniques (Debar, et al., 1992) (Ryan, et al., 1998) (Mukkamala, et al., 2002).
The major differences in the various techniques are based on the learning algorithms used and the NN structure.

The large variety of different solutions to ID pose the problem of determining which solution is best suited to the given environment. To aid in this task the following table was created, summarising the strengths and weakness of the three major ID techniques.

<table>
<thead>
<tr>
<th>Requires time consuming setup</th>
<th>Misuse Detection</th>
<th>Anomaly Detection</th>
<th>Specification Detection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requires time consuming setup</td>
<td>Yes</td>
<td>No</td>
<td>No (in cases with well defined behaviours)</td>
</tr>
<tr>
<td>High rate of false alarms</td>
<td>No</td>
<td>Yes (can be reduced)</td>
<td>No</td>
</tr>
<tr>
<td>Computationally expensive</td>
<td>No</td>
<td>Yes (in most cases)</td>
<td>No</td>
</tr>
<tr>
<td>System Dependent</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Detection against unseen attacks</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 1 Summary of ID techniques

Each technique has a variety of approaches and each approach its’ own design choices. Extensive research was thus required to make an informed decision for the ID module design.

4 ID Module

The ID module is tasked with detecting attack behaviours from a given set of events. As previously mentioned this is a key component to the Anomalous system and as such is vital to its performance.

4.1 Requirements

The performance of the ID module can be measured by a few key characteristics: accuracy, scalability and adaptability.

Accuracy is the measure of how many of the classifications made by the ID module are correct. Accuracy is dependent on three major classification types:

- True positives (TP) which are anomalies which are correctly classified as anomalies by the ID module
- False positives (FP) which are normal behaviours incorrectly classified as anomalies by the ID module
- False negatives (FN) which are anomalies incorrectly classified as normal by the ID module.

In order to have an accurate ID module we need to maximise the TP whilst minimising the FP and FN.

The accuracy of the system is also dependant on the data used to train the system, as such we also need to ensure that the correct input data is used. It’s also vital to ensure this data is interpreted correctly and efficiently, so as to not skew the detection process.

The scalability of the ID module is dependent on how it copes with increasing data sizes. This is important in ensuring the anomalous system copes well with target environments that produce large amounts of data. The ID module thus has to be able to process large amounts of data in as fast a time as possible, and in a manner such that computational time does not increase exponentially with data size.

The adaptability of the Anomalous system is also important in ensuring it can cope with different target environments. To make the Anomalous system adaptable we need an ID
module which can adapt to different target environments. Adaptability is however not an empirically measurable characteristic, so in order to ensure the ID module remains adaptable we need to make design choices which will result in the least effort required to adapt the ID module to a new target environment. These design choices still need to satisfy the accuracy and scalability requirements. Scalability is a factor in determining how adaptable a system is and as such satisfying the scalability requirement will help satisfy the adaptability requirement. On the contrary, accuracy is often sacrificed in order to increase adaptability (Debar, et al., 1999). As such a compromise needs to be reached to allow for high levels of accuracy without leaving the system requiring lots of adaptation for new target environments.

The ID module also has an effect on the collection and visualisation modules. The data required for input to the ID module must be collected by the collection module. A large list of input features (fields of data) for each event would result in a large network traffic overhead between the modules and a larger overhead for data collection. The ID module must thus seek to minimise the data needed for learning and detection, without significantly compromising accuracy.

Once the ID module processes the data the outputs are used by the visualisation module for display to system administrators. The outputs thus need to be informative and accurate. Accuracy is a defacto requirement of the ID module, therefore aiding the visualisation module. For the visualisation to be of greatest use to the system administrators, it needs to provide them with as much information on suspected anomalies as possible. Thus the ID module must seek to provide the visualisation module with as much relevant information as possible. Too many FPs can also inhibit the performance of the visualisation module by cluttering the display and wasting the time of system administrators as they have to sift through the false alarms.

5 ID Module Design

The goal of the ID module, within the Anomalous system is to detect attack behaviours. This allows system administrators to better respond to and protect their systems from attacks.

The design of the ID module resulted in extensive research, which was needed to facilitate the various design choices. The researched showed that the target environment for the ID module is very important (Ryan, et al., 1998). Therefore we must first re-asses the context of the ID module before we can create a design to fulfil the ID module requirements.

5.1 Context

The final goal of the Anomalous system will be a useful and complete system, which would be suited for deployment in many environments. This goal was kept in mind when making design choices for the ID module.

![Diagram illustrating the interactions between the three modules of the Anomalous system](image_url)

When designing the Anomalous ID module we must consider the intended environment in
which the ID module will operate. In the *Anomalous* system environment the ID module is assisted by the collection and visualisation modules.

The collection module allows for us to focus on the design of the ID module without having to worry about how the data is collected. It is however important to consider the implications of the ID module on the performance of the system as a whole. As such we need to limit the data required by the ID module as much as possible to allow for scalability of the collection module and in the *Anomalous* system as a whole.

The visualisation module deals with the outputs of both the collection and ID modules and as such we need to design an ID module which will provide useful classifications to the visualisation module. These classifications should allow the end users (system administrators) to better utilise the *Anomalous* system. For this we need classifications which are accurate and informative.

To ensure accuracy we seek to reduce the number of FN as a prime concern, so that the *Anomalous* system does not miss any attacks. A secondary concern is reducing the number of FP as this will help the system administrators process the output of the system quicker, as they won’t have to waste time investigating activity which is actually normal activity. Another design feature which could allow the Visualisation module to provide more useful information to the system administrators would be detailed event classification. This would allow the system administrators to see what type of anomaly has been detected, rather than just if behaviour is anomalous or normal. This would aid the system administrators in determining the cause and effects of the attack behaviour as well as identifying if it was a FP or not.

Due to the requirements of the anomalous system, we have decided, after lots of research, to use a NN based approach for the ID module.

5.2 Reasons for choosing Neural Network based approach

NN provide a set of key features which make them best suited to the *Anomalous* system. Through extensive research we have identified the following key features which motivated our choice to use NN for the ID module of the *Anomalous* system.

As mentioned by Mukkamala et al. one of the advantages provided by NN is that they can provide multi-category classifications, which is a key consideration in the ID module design (Mukkamala, et al., 2002). This NN based design will allow us to classify the behaviour into one of 6 main categories:

1. Normal behaviour
2. DOS: Denial of service (attack)
3. R2L: unauthorized access from a remote location (attack)
4. U2R: unauthorized access to local super user/ root (attack)
5. Probing: surveillance or other probing (attack)
6. Other unknown attack

The 4 known attack classes can then be further broken down into 32 different known exploits to provide detailed information to the Visualisation module.

In order to be effective in different environments the *Anomalous* system needs to be adaptable. Thus the ID module needs to be designed such that it can be used for a new target environment with as little work as possible required to adapt it to the new environment. NN can provide this facility.

The adaptability of NN, as highlighted by Garcia-Teodoro et al., make them uniquely
suited to this challenge and is thus one of the motivating factors for choosing NN as the anomaly detection method for the ID module (García-Teodoro, et al., 2009).

Another key feature which aids in the adaptability of the system is a NN's ability to learn new behaviours once they are identified (Sung, et al., 2003). This can be done by re-running the learning process once significant new behaviours are found. This helps to keep the ID module up to date and helps it to cope with changing environments (García-Teodoro, et al., 2009).

NN's have been extensively used in the Intrusion Detection community, (Ibrahim, 2010) (Mukkamala, et al., 2002) (Ryan, et al., 1998) (Debar, et al., 1992), and can thus be seen to be proven to work for the task required by the ID Module. This is important as a reliable ID module is required for the success of the Anomalous system.

5.3 Neural Network Design
All NN for Intrusion Detection have a similar defining structure. These NN are composed of three layers, each with varying numbers of nodes and connections between the nodes.

The input layer relates the inputs of the system into a format which can be understood and processed by the NN. The input layer is important for correctly interpreting the data; it is thus important to correctly pre-process and represent the data used for the input layer.

The hidden layer is where the majority of processing and classification happens. The structure of the hidden layer is therefore usually different for each different problem.

The output layer provides a means of outputting the results of the NN classification. It is thus important that the output layer provide a method for detailed classification.

These three layers can be connected in a variety of ways, in the chosen feed forward network each layer is connected to the layer before it by a set of unidirectional weights. When the weights are combined with the values of the previous neurons and summed across all inputs to that neuron, a value for the neuron is provided. This process is used to determine the values for each Neuron, for each input, and as such allows classification to be done at the output layer, based on a certain threshold value.

5.3.1 Inputs
One of the most difficult aspects of applying a NN to Event based Anomaly Detection is adapting the data for processing by the NN (Sung, et al., 2003). The data in event logs is often in a non-machine readable form which means it needs to be pre-processed before it can be used by the NN (Mukkamala, et al., 2002).

A new representation for events needs to be created in order to allow them to be processed by the NN. This representation needs to be a detailed but efficient representation that avoids capturing useless
data but also ensures that all the important data is included (Sung, et al., 2003).

In order to find a data representation which was suitable to the Anomalous system we did extensive research into different input representations (Sung, et al., 2003) (Mukkamala, et al., 2002) (Ibrahim, 2010) (Garcia-Teodoro, et al., 2009) (Peddabachigari, et al., 2007).

This extensive research revealed that the best suited representation is a list of 41 features which accurately and efficiently describe the events (Refer to Appendix A).

As shown by Sung et al. the input features for NN can be reduced to a list of 34 important features without a significant loss of accuracy. They found that although there was a significant increase in FPs (6.66% to 18.19 %) there was also a drastic reduction in FNs (6.27% to 0.25%) and training time (412 epochs to 27 epochs) (Sung, et al., 2003). These results show that reducing the feature list (and hence the input list) can be beneficial in ensuring less FNs, which can be more harmful than FPs, and allowing for greater scalability due to the greatly reduced learning time. The reduced feature list also requires the Collection module to collect and store less data, thereby also enhancing the scalability of the system as a whole.

In order to use this feature list we needed an appropriate data source. Due to the nature of the data provided for this project by our supervisor Andrew Hutchison, which lacked classifications needed for training and testing, we need another data source which can provide the required classifications. As such we discovered a well used source of data, which is from the 1998 DARPA KDD cup which is standard benchmark data for Intrusion Detection (Sung, et al., 2003). This data was obtained by creating a simulation of a US Air Force base LAN and recording the raw TCP/IP dump data. This LAN was run like a true environment, but being blasted with multiple attacks. The previously mentioned 41 features were then extracted from the data and the behaviour classifications were added for learning and testing purposes.

More recent research (Tavallaee, et al., 2009) has shown that the KDD CUP 99 data set has certain deficiencies. As such a variant of this data was chosen which does not have any of the discovered deficiencies. This is the NSL-KDD data set proposed by Tavallaee et al. (Tavallaee, et al., 2009). This has thus been chosen as the data set to be used for learning and testing.

If we were to have used the data which was provided to us, extensive pre-processing would have needed to be done using automated parsers such as SNMP (Goldstein, et al., 2012). This is however an unnecessary overhead due to the existence of already formatted and standardised data. Using the NSL-KDD data also allows us to perform comparisons with existing systems (Mukkamala, et al., 2002) and use research based on this data (Sung, et al., 2003) (Tavallaee, et al., 2009) to further improve the ID module design.

The 34 important features which were identified will determine the number of input nodes in the NN. Thus our NN for the ID module will have 34 inputs, one for each important feature.

Another important factor to consider is that each event by itself may not be an attack, but it is a sequence of events which constitutes an attack. These sequences also varying depending on the type of attack (Sung, et al., 2003). This is an especially difficult challenge to overcome and requires the NN to look at a window of events to determine if they are...
part of an attack. It is important not to make the window too large as this will result in some smaller attack signatures going unseen, however if the window is too small then the larger sequences could be incorrectly classified (Biermann, et al., 2001).

A window size of 3 on the following sequence of calls:

\textit{open, read, write, close, close}

the following table of windows would be observed:

<table>
<thead>
<tr>
<th>open</th>
<th>read</th>
<th>write</th>
</tr>
</thead>
<tbody>
<tr>
<td>read</td>
<td>write</td>
<td>close</td>
</tr>
<tr>
<td>write</td>
<td>close</td>
<td>close</td>
</tr>
</tbody>
</table>

Optimal window size differs depending on the problem. Due to the nature of the NSL-KDD data the events are no longer listed in a strict ordering due to the random sampling used to create the various training and learning data sets. This makes the use of window sizes for the ID module obsolete. Part of the design of the NSL-KDD data set is to overcome the sequencing issues with ID, and therefore we can be confident in using a single input at a time during learning and detection (Tavallaee, et al., 2009).

### 5.3.2 Hidden Layer

The Hidden Layer of the NN is designed according to the specific problem. The design choices that need to be made are how many hidden layers there will be, and how many neurons will be in each layer.

Mukkamala et al. have shown in their experiments, based on the KDD CUP 99 data set, that the best structure for the hidden layer is to have two layers of 40 neurons each. They found that this structure provided the best detection accuracy and training time (Mukkamala, et al., 2002).

This 40-40 structure is well suited to avoiding the problems of over-fitting and under-fitting as the hidden layer is complex enough, without being overly complex (Barry, 2000).

### 5.3.3 Output

The output layer represents the results of the NN classification. The most important classification we are seeking to achieve is whether or not the observed inputs (events) constitute anomalous behaviour.

Classification according which type of attack is suspected is useful but of secondary importance.

<table>
<thead>
<tr>
<th>Attack Class</th>
<th>OS: Solaris</th>
<th>OS: SunOS</th>
<th>OS: Linux</th>
</tr>
</thead>
<tbody>
<tr>
<td>Denial of Service</td>
<td>Apache2</td>
<td>Apache2</td>
<td>Apache2</td>
</tr>
<tr>
<td></td>
<td>Back</td>
<td>Mail bomb</td>
<td>Back</td>
</tr>
<tr>
<td></td>
<td>Mail bomb</td>
<td>Neptune</td>
<td>Mail bomb</td>
</tr>
<tr>
<td></td>
<td>Neptune</td>
<td>Ping of death</td>
<td>Neptune</td>
</tr>
<tr>
<td></td>
<td>Ping of death</td>
<td>Process table</td>
<td>Ping of death</td>
</tr>
<tr>
<td></td>
<td>Process table</td>
<td>Smurf</td>
<td>Process table</td>
</tr>
<tr>
<td></td>
<td>Smurf</td>
<td>Sydoslog</td>
<td>Smurf</td>
</tr>
<tr>
<td></td>
<td>Sydoslog</td>
<td>UDP storm</td>
<td>Sydoslog</td>
</tr>
<tr>
<td></td>
<td>UDP storm</td>
<td></td>
<td>UDP storm</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Remote to User</th>
<th>Dictionary</th>
<th>Ftp-write</th>
<th>Guest</th>
<th>Php</th>
<th>Xlock</th>
<th>Xsnnoop</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Dictionary</td>
<td>Ftp-write</td>
<td>Guest</td>
<td>Php</td>
<td>Xlock</td>
<td>Xsnnoop</td>
</tr>
<tr>
<td></td>
<td>Ftp-write</td>
<td>Dictionary</td>
<td>Guest</td>
<td>Php</td>
<td>Xlock</td>
<td>Xsnnoop</td>
</tr>
<tr>
<td></td>
<td>Guest</td>
<td>Ftp-write</td>
<td>Dictionary</td>
<td>Guest</td>
<td>Php</td>
<td>Xlock</td>
</tr>
<tr>
<td></td>
<td>Php</td>
<td>Ftp-write</td>
<td>Dictionary</td>
<td>Guest</td>
<td>Php</td>
<td>Xlock</td>
</tr>
<tr>
<td></td>
<td>Xlock</td>
<td>Ftp-write</td>
<td>Dictionary</td>
<td>Guest</td>
<td>Php</td>
<td>Xlock</td>
</tr>
<tr>
<td></td>
<td>Xsnnoop</td>
<td>Ftp-write</td>
<td>Dictionary</td>
<td>Guest</td>
<td>Php</td>
<td>Xlock</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User to Super user</th>
<th>Eget</th>
<th>Fbcoxfig</th>
<th>Fdformat</th>
<th>Ps</th>
<th>Perl</th>
<th>Xterm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Eget</td>
<td>Fbcoxfig</td>
<td>Fdformat</td>
<td>Ps</td>
<td>Perl</td>
<td>Xterm</td>
</tr>
<tr>
<td></td>
<td>Eget</td>
<td>Fbcoxfig</td>
<td>Fdformat</td>
<td>Ps</td>
<td>Perl</td>
<td>Xterm</td>
</tr>
<tr>
<td></td>
<td>Eget</td>
<td>Fbcoxfig</td>
<td>Fdformat</td>
<td>Ps</td>
<td>Perl</td>
<td>Xterm</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Probing</th>
<th>Ip sweep</th>
<th>Mscan</th>
<th>Nmap</th>
<th>Satan</th>
<th>Ip sweep</th>
<th>Mscan</th>
<th>Nmap</th>
<th>Satan</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ip sweep</td>
<td>Mscan</td>
<td>Nmap</td>
<td>Satan</td>
<td>Ip sweep</td>
<td>Mscan</td>
<td>Nmap</td>
<td>Satan</td>
</tr>
<tr>
<td></td>
<td>Ip sweep</td>
<td>Mscan</td>
<td>Nmap</td>
<td>Satan</td>
<td>Ip sweep</td>
<td>Mscan</td>
<td>Nmap</td>
<td>Satan</td>
</tr>
</tbody>
</table>

Table 2: Attack types for the NSL-KDD data

The NN structure for output is easy to define as the standard design is to have a single output neuron (Mukkamala, et al., 2002). We will then use a class index in order to distinguish which class of classification the output belongs to.

The results of the NN classification are important in providing the Visualisation module with accurate information. The system administrators that use the Visualisation component will be able to detect some FPs based on the displayed information in the Visualisation component (such as false
positive for a DOS attack may be much smaller than other nodes where true DOS attacks are originating from, this can be seen by the size of the nodes in the Visualisation). Thus the Visualisation component helps reduce the effect of FPs on the usefulness of the Anomalous system. This allows the ID module to focus on ensuring a lower degree of FNs through the techniques mentioned in the Input subsection and detailed by Sung et al. (Sung, et al., 2003).

5.4 Learning
The learning process of a NN is important in ensuring an effective and robust system. In NN the learning process consists of adjustments made to the parameters, representing the weights of edges, of the NN in order to achieve results congruent with those expected for a set of chosen events. The outputs of the NN for the set of chosen events, with already known outputs, are compared to the outputs of the NN in order to make adjustments to the parameters of the NN and achieve the expected outputs.

In order to allow for effective comparison between existing solutions the extensively used Back Propagation algorithm will be used for the learning process of the NN (Ryan, et al., 1998).

The Back Propagation algorithm uses the calculated difference in target outputs to generated outputs and adjusts the weights to fit the target. This is repeated for all the layers, starting from the output layer. This algorithm relies upon the use of differentiation and thus requires the use of a differentiable activation function, further motivating our choice of using the sigmoid function as the activation function (Hecht-Nielsen, 1989).

The Back Propagation algorithm uses a learning rate factor to determine how fast the weights of the NN change. Amini has shown that for a similar problem and a 40-40 hidden layer structure, the optimal learning rate is between 0.0001-0.0006 (Amini, 2008). Thus we shall use a learning rate of 0.0004.

![Figure 8: Illustration of the Back Propagation algorithm, the solid lines show the weights and the forward propagation, whilst the dotted lines show the backwards propagation of errors.](image-url)

In order to train the NN a special data set is needed for the learning process. This data set needs to have labels which indicate the correct classifications for the data. For this purpose we chose to use the NSL-KDD data set which has classification labels.

The NSL-KDD data set is preferred to the standard KDD CUP 99 data set because of the issues highlighted by Tavallaee et al., which would negatively affect the quality of the learning process using the KDD CUP 99 data (Tavallaee, et al., 2009).

Only a subset of the full NSL-KDD data set will be used for learning. This subset does not have all the attack types included, so as to test the NN against unseen attacks.
5.5 Detection
In order to detect anomalies the ID module takes a list of inputs representing the behaviour of the network, these inputs are then processed by the ID module which will output a classification for each given input.

The value of a specific neuron is determined by a summation of weights leading to a neuron (and the values of the previous connected neurons). Edge weight is multiplied by the neuron from which the connection originates, and this is used in the summation. This is repeated until the values are calculated for the output layer and from these values classifications are made. Edge weights are defined by the learning process.

The following figure illustrates a simple case of NN detection. This simple NN performs the XOR operation through a simple NN structure and weighing scheme.

The classification is determined based on a threshold value for the output which is adjusted during the learning process.

5.6 Code Structure
In order to propose a NN design which is well suited to the Anomalous system we need to make certain design choices in regards to the coding of the NN. We need an ID module which can be deployed across multiple platforms. For this reason we propose Java as the language for development.

Furthermore we propose the use of an Object Orientated Programming (OOP) in order to allow for easy adaptation of the ID module between different target environments. Performance defining variables such as NN structure have also been made easily changeable. These two factors allow for changes to be easily made to the system as research reveals new findings and environments change. This allows the Anomalous system to remain current and effective.

Extensive discussions (with developers of the Collection and Visualisation modules) have also identified key requirements and design choices to allow for easy future integration of the various modules that make up the Anomalous system and any possible future modules. We have identified and chosen the ways in which the modules would communicate so as to minimise integration effort. We have chosen to use standardised Database (DB) structures and management systems in order to facilitate smooth communication and integration of modules.

5.7 Evaluation
There are 2 major factors which we can analyse and evaluate, accuracy and scalability.
Accuracy is dependent on 3 key factors, True Positives (TP), FP and FN. We want high TP and low FN and low FP. We will define accuracy the percentage of correctly classified events. This can be written as:

\[
\frac{\text{TP}}{\text{True anomalies}}
\]

where True anomalies represents the number of actual anomalous events in the test data set. We will also calculate the percentage of FP and FN to enable us to compare the results to previous research results.

In order to empirically test accuracy we propose the use of a set of data points from the NSL-KDD data set, which was not a part of the learning set. This test set will also contain some attacks not in the learning data set, in order to allow us to test against unseen attacks.

Scalability can be seen in both the context of learning and detection time. Learning scalability is less important as this can be done offline as the Anomalous system is being set up. Detection scalability is however more crucial to the operation of the Anomalous system. Detection needs to be done in a time critical manner to allow for fast and efficient use of the Anomalous system by the end users.

Scalability can be tested by running the learning and detection on increasing amounts of data. This is done by taking progressively larger portions of the NSL-KDD data set and comparing learning and detection times for the different data sets.

There are other research goals which we are not as easily testable as accuracy and scalability.

A goal of the ID module was to be adaptable to different environments. This is not easily tested as we would require multiple target networks with appropriately labelled data. Many of our design choices were however motivated with this goal in mind, as such the system should prove easy to adapt to new target environments. A possible way of testing this would be to deploy the Anomalous system in multiple environments and record the number of changes to the ID module required (to achieve sufficient performance) for each target environment. This would allow us to see how well the ID module copes with new environments and how much effort would be required to achieve sufficient performance for a given environment.

We have designed the ID module to allow it to be fairly and easily compared with existing designs (Mukkamala, et al., 2002) (Sung, et al., 2003).

5.8 Implementation

This project has focused on the design of and ID module for use in the Anomalous system. The implementation itself is therefore left to future work.

In order to determine the feasibility of many of the choices made during design prototypes were created to simulate the various aspects of the ID module.

One such prototype was a parser which successfully processes and effectively formats the NSL-KDD data for use in the NN. This parser has been tested to successfully parse data from the testing and learning NSL-KDD data sets.

A multi-layered feed forward neural network structure was created to easily facilitate the creation of any given structure of NN. This helped to inform many of the decisions made regarding the structural design of the NN.

Attempts at implementing a variety of learning algorithms were made, in order to
better inform the decision of which algorithm was best suited for the ID module.

The major problem encountered during implementation and design was the sheer amount of research needed to be done. As this was the first time we have dealt with NN, or any real machine learning, a lot of research was required to gain a deep enough understanding of all the nuances of NN and other ID techniques.

Much research went into investigating other ID techniques, in order to equip ourselves with enough knowledge to make informed design decisions.

The lack of knowledge of the nuances of the ID problem and the context specific challenges was highlighted by the changes in design that occurred during the course of this project. The chosen ID approach was changed from a genetic algorithm approach (Bridges, et al., 2000) to a NN approach to facilitate scalability and reduce learning times. The chosen learning algorithm for the NN was also changed to allow for fairer comparison with other systems and faster training convergence (Lei, et al., 2011) (Sung, et al., 2003).

The target task was also changed from a network based ID system to an Event based ID system. This required changes in the algorithms used as well as the data formatting and input features.

Another change was from the supplied data to the NSL-KDD data, which was required due to the lack of classification labels need for learning and testing in the supplied data. This resulted in design changes for the input layer of the NN, and the methods for processing the data.

Although the final product of this project is the design itself, the implemented parts of the ID module have allowed for a more informed and realistic design.

5.9 Integration
Integration of the various modules of the Anomalous system was a key issue considered in the design and implementation of the ID module. However the integration process itself was defined as future work, to be done at a later date. This integration should be a fairly smooth process due to the nature of the design and implementation.

Many design aspects such as output formats and database systems were carefully chosen together with the designers of the Visualisation and Collection modules to allow for the modules of the Anomalous system to be easily integrated.

6 Discussion
The result of this project is the design proposed for the ID module.

Although parts of the design have been implemented and tested for feasibility the full design would need to be implemented to empirically test the accuracy, scalability and adaptability of the system.

We can analyse the design to ascertain whether or not it fulfils the aims and requirements set out for the ID module.

The accuracy of the algorithms used in the design, with the input used in the design, has been shown to achieve acceptable levels of accuracy (Sung, et al., 2003). The learning and detection algorithms, and the data format chosen for the design are well used for ID systems, proving they will provide sufficient accuracy (Hecht-Nielsen, 1989) (Debar, et al., 1992) (Tavallaee, et al., 2009) (Sung, et al., 2003). It is this large quantity of evidence which motivated the design choices and
ultimately shows that the design of the ID module will provide sufficient accuracy.

In order to ensure the ID module can scale with large input data we have designed a system which reduces duration of the most costly step, name the learning. As shown by Sung et al. the reduction of the input features needed by the module will drastically reduce the time taken to complete the learning process (Sung, et al., 2003). This input reduction will also result in less traffic required by the ID module, aiding the Anomalous system in achieving a scalable solution.

The learning for the ID module is the most time intensive process but only needs to be done periodically. An initial learning process needs to be run for a new environment, after which period updates can be done to the ID module with any newly discovered patterns. This allows the ID module learning to not negatively affect the scalability of the system when in use by system administrators.

During use of the system, the detection phase is the most time consuming phase of the ID module. The detection phase constitutes a single forward propagation of the NN for each input. Thus the complexity is:

\[ n \times \text{layers} \times \text{neurons per layer} \times \text{weights per neuron} \]

Where \( n \) is the number of inputs, which is linear \( \Theta(n) \) as the number of weights, neurons and layers is constant.

The detection, which is the most time critical task is thus linearly scalable, which is optimal for the problem as each input must be considered and classified.

Although adaptability is difficult to measure in empirical terms (and beyond the scope of this project) we can look at the effort which is likely to be required to adapt the ID module of the Anomalous system to a new environment.

The Anomalous system provides a great deal of adaptive power. With the relevant training data, the ID module can relearn the behaviour of a new set of events in a different environment.

The number of changes required for a new environment is thus far less than the effort which would be required to redesign the ID module, as would be required by various other approaches, such as misuse detection (Debar, et al., 1999). Thus the system provides a satisfactory level of adaptability to new environments without incurring large costs in terms of effort.

The number of changes which would need to be made will also have diminishing returns as each change will likely only decrease in its' affect on the overall accuracy of the system as more changes are made.

The design of the ID module consists of a combination of proven approaches which together will be best suited for use within the Anomalous system. This combination of proven approaches yields a design which should prove to be feasible and successful.

7 Conclusions

In this paper we have proposed an ID module design which is feasible. This design has come about through extensive research and prototyping.

Various algorithmic approaches for the ID module have been investigated through lots of careful research and various other papers have proven that some of these approaches are successful in very similar contexts.

The design of the ID module is context specific, in that it is intended for use in the
Anomalous system. This context has motivated many of the design choices and will provide a robust and well suited ID module. The integration with the other modules in the Anomalous system will provide a useful, full package, Intrusion Detection and analysis system.

The ID module was designed to be adaptable to multiple environments, allowing the Anomalous system to be deployed in any given environment with minimal effort.

We have shown that the proposed design fulfils the aims and requirements set out for the ID module, and provides a blueprint for possible future implementations in systems such as Anomalous.

8 Future work
The primary concern for any future work on the Anomalous system should be implementing and integrating the collection, ID and visualisation modules. This should be a smooth process due to the careful design choices which were made with future implementation and integration in mind.

In terms of the ID module, there are various aspects which could be improved upon in future iterations. Optimisations could be made to various aspects of the ID module.

The NN learning and detection process could be optimised through the use of parallelisation methods. Care must be taken when parallelising the learning algorithm as if certain steps are done out of order then the efficacy of the learning process may suffer. On the other hand the detection process can be easily parallelised as each classification can be done independently of the other classifications.

As shown by Heaton, backpropagation can be done using simple Java multithreading on a CPU, with sufficient performance enhancements in terms of learning time (Heaton, 2009).

Lopes et al. have also shown that backpropagation can be implemented using CUDA on a GPU, providing reductions in computational cost (Lopes, et al., 2009).

As has been shown by Sung et al. further reductions can be done on the input feature list (Sung, et al., 2003). These reductions could further improve performance. Care must be taken not to significantly negatively affect the accuracy of the ID module.

Comparisons can be made between the NN approach and other approaches to the given problem and environment. This will show which approach is best suited to the Anomalous system. To facilitate fair comparison the new approaches must be tested within the same context as the NN approach adopted in this report.

The results for the NN approach, once implemented, can be compared to other anomaly detection methods, such as Support Vector Machines (Mukkamala, et al., 2002) or genetic algorithms (Bridges, et al., 2000).

Comparisons can also be made with other ID techniques, such as specification detection (Tseng, et al., 2006) or misuse detection (Kumar, et al., 1994).

The learning algorithm used in this NN implementation could also be compared to other learning algorithms and possibly approved upon (Lei, et al., 2011).

Using backpropagation for learning can result in the learning process becoming stuck in local minima. As shown by Gori et al. we can ensure the convergence to an optimal solution by the use of batch learning (M. Gori, 1992). In batch learning many propagations...
occur before the weight updating occurs. This future development would help to make the system more robust.

Tests could be done in terms of optimisation of the code structure; a non OOP version could be compared to the current code structure design to see how the OOP affects the performance of the system.

Another important set of tests that could be done on the system are tests involving updated real world data. The *Anomalous* system could also be deployed in a real world context to empirically measure the systems performance and determine which areas require major improvement.

There is lots of future work that could be done to improve the *Anomalous* system and the ID module to ensure that the system will be useful within the context of today’s ever changing Information Technology environment. An example of this could be adapting the system for use in a cloud computing context which would allow for the system to further service multiple target networks in an efficient and sustainable manner.

All these possibilities for future improvements demonstrate the versatility of a system such as the *Anomalous* system. The variety of improvements also highlights the various avenues of growth for the system, which would ensure it would become a useful tool for network security.

9 Acknowledgements

We would like to thank Dr. Michelle Kuttel and Dr Andrew Hutchison for their guidance on this project. Thanks must also go to Dr. Andrew Hutchison for providing us with real world data for use in development of the *Anomalous* system. We would also like to thank Dr. Geoff Nitschke for his advice and assistance on matters pertaining to the ID problem and the NN development.

Thanks must also go out to the tireless work of the many people who create useful NN based tutorials, such as (Jain, et al., 1996).
Works Cited


### Table 3 Full 41 input feature list

<table>
<thead>
<tr>
<th>feature name</th>
<th>description</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>duration</td>
<td>length (number of seconds) of the connection</td>
<td>continuous</td>
</tr>
<tr>
<td>protocol_type</td>
<td>type of the protocol, e.g. tcp, udp, etc.</td>
<td>discrete</td>
</tr>
<tr>
<td>service</td>
<td>network service on the destination, e.g., http, telnet, etc.</td>
<td>discrete</td>
</tr>
<tr>
<td>src_bytes</td>
<td>number of data bytes from source to destination</td>
<td>continuous</td>
</tr>
<tr>
<td>dst_bytes</td>
<td>number of data bytes from destination to source</td>
<td>continuous</td>
</tr>
<tr>
<td>flag</td>
<td>normal or error status of the connection</td>
<td>discrete</td>
</tr>
<tr>
<td>land</td>
<td>1 if connection is from/to the same host/port; 0 otherwise</td>
<td>discrete</td>
</tr>
<tr>
<td>wrong_fragment</td>
<td>number of <code>wrong</code> fragments</td>
<td>continuous</td>
</tr>
<tr>
<td>urgent</td>
<td>number of urgent packets</td>
<td>continuous</td>
</tr>
<tr>
<td>hot</td>
<td>number of <code>hot</code> indicators</td>
<td>continuous</td>
</tr>
<tr>
<td>num_failed_logins</td>
<td>number of failed login attempts</td>
<td>continuous</td>
</tr>
<tr>
<td>logged_in</td>
<td>1 if successfully logged in; 0 otherwise</td>
<td>discrete</td>
</tr>
<tr>
<td>num_compromised</td>
<td>number of <code>compromised</code> conditions</td>
<td>continuous</td>
</tr>
<tr>
<td>root_shell</td>
<td>1 if root shell is obtained; 0 otherwise</td>
<td>discrete</td>
</tr>
<tr>
<td>su_attempted</td>
<td>1 if <code>su root</code> command attempted; 0 otherwise</td>
<td>discrete</td>
</tr>
<tr>
<td>num_root</td>
<td>number of <code>root</code> accesses</td>
<td>continuous</td>
</tr>
<tr>
<td>num_file_creations</td>
<td>number of file creation operations</td>
<td>continuous</td>
</tr>
<tr>
<td>num_shells</td>
<td>number of shell prompts</td>
<td>continuous</td>
</tr>
<tr>
<td>num_access_files</td>
<td>number of operations on access control files</td>
<td>continuous</td>
</tr>
<tr>
<td>num_outbound_cmds</td>
<td>number of outbound commands in an ftp session</td>
<td>continuous</td>
</tr>
<tr>
<td>is_hot_login</td>
<td>1 if the login belongs to the <code>hot</code> list; 0 otherwise</td>
<td>discrete</td>
</tr>
<tr>
<td>is_guest_login</td>
<td>1 if the login is a <code>guest</code> login; 0 otherwise</td>
<td>discrete</td>
</tr>
<tr>
<td>count</td>
<td>number of connections to the same host as the current connection in the past two seconds</td>
<td>continuous</td>
</tr>
<tr>
<td>serror_rate</td>
<td>% of connections that have <code>SYN</code> errors</td>
<td>continuous</td>
</tr>
<tr>
<td>rerror_rate</td>
<td>% of connections that have <code>REJ</code> errors</td>
<td>continuous</td>
</tr>
<tr>
<td>same_srv_rate</td>
<td>% of connections to the same service</td>
<td>continuous</td>
</tr>
<tr>
<td>diff_srv_rate</td>
<td>% of connections to different services</td>
<td>continuous</td>
</tr>
<tr>
<td>srv_count</td>
<td>number of connections to the same service as the current connection in the past two seconds</td>
<td>continuous</td>
</tr>
<tr>
<td>srv_serror_rate</td>
<td>% of connections that have <code>SYN</code> errors</td>
<td>continuous</td>
</tr>
<tr>
<td>srv_rerror_rate</td>
<td>% of connections that have <code>REJ</code> errors</td>
<td>continuous</td>
</tr>
<tr>
<td>srv_diff_host_rate</td>
<td>% of connections to different hosts</td>
<td>continuous</td>
</tr>
<tr>
<td>dst_host_count</td>
<td>destination host count</td>
<td>continuous</td>
</tr>
<tr>
<td>dst_host_srv_count</td>
<td>destination host server count</td>
<td>continuous</td>
</tr>
<tr>
<td>dst_host_same_srv_rate</td>
<td>destination host same server rate</td>
<td>continuous</td>
</tr>
<tr>
<td>dst_host_diff_srv_rate</td>
<td>destination host different server rate</td>
<td>continuous</td>
</tr>
<tr>
<td>dst_host_same_src_port_rate</td>
<td>destination host same source port rate</td>
<td>continuous</td>
</tr>
<tr>
<td>dst_host_srv_diff_host_rate</td>
<td>destination host server different host rate</td>
<td>continuous</td>
</tr>
<tr>
<td>dst_host_serror_rate</td>
<td>destination host SYN error rate</td>
<td>continuous</td>
</tr>
<tr>
<td>dst_host_srv_serror_rate</td>
<td>destination host server SYN error rate</td>
<td>continuous</td>
</tr>
<tr>
<td>dst_host_rerror_rate</td>
<td>destination host REJ error rate</td>
<td>continuous</td>
</tr>
<tr>
<td>dst_host_srv_rerror_rate</td>
<td>destination host server REJ error rate</td>
<td>continuous</td>
</tr>
</tbody>
</table>